Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 57 (2015) 222 – 231

Procedia Computer Science

3rd International Conference on Recent Trends in Computing (ICRTC-2015)

An Improved Approach to Maximize the Performance of Disk Scheduling Algorithm by Minimizing the Head Movement and Seek Time using Sort Mid Current Comparison (SMCC) Algorithm

Mahesh Kumar M R[a,*], Renuka Rajendra B[b]

[a]*Assistant Professor, Dept. of CSE, JSSATE, Bengaluru, 560060, India*
[b]*Assiatant Professor, Dept of CSE, JSSATE, Bengaluru, 560060, India*

Abstract

One of the main goal of the operating system for the disk drives is to use the hardware efficiently. we can meet this goal using fast access time and large disk bandwidth that depends on the relative positions of the read-write head and the requested data. Since memory management allows multiprogramming so that operating system keeps several read/write request in the memory. In order to service these requests, hardware (disk drive and controller) must be used efficiently. To support this in disk drive, the hardware must be available to service the request. if the hardware is busy, we can't service the request immediately and the new request will be placed in the queue of pending requests. Several disk scheduling algorithms are available to service the pending requests. among these disk scheduling algorithms, the algorithm that yields less number of head movement will remain has an efficient algorithm.

In this research paper, we propose a new disk scheduling algorithm that will reduce the number of movement of head thereby reducing the seek time and it improves the disk bandwidth for modern storage devices. Our results and calculations show that, proposed disk scheduling algorithm will improve the performance of disk i/o by reducing average seek time compared to the existing disk scheduling algorithm. For few requests, the seek time and the total number of head movement is equal to SSTF or LOOK scheduling.

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)

Keywords : Seek time ; Transfer time ; disk bandwidth ; head movement ; disk scheduling;

* Mahesh Kumar M R. Tel.: +91 9916616898;
E-mail address: maheshkumar.027@hotmail.com

Peer-review under responsibility of organizing committee of the 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
doi:10.1016/j.procs.2015.07.468

1. Introduction

There are several disk scheduling algorithms (FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK) are available to service the pending requests in the queue. the most widely known algorithm for scheduling the request is SSTF or LOOK algorithm. the main goal of this and most of other disk scheduling algorithm is to reduce the number of disk head movement thereby reducing the seek time. To achieve this, we should have a fast access time and large disk bandwidth. over the years the rotational speed of disks has increased only slightly, while the full stroke seek time has been shortened significantly [1].

Data is recorded on a series of magnetic disks or platters, connected to a spindle that rotates at high speed [3]. A read-write head lies just above the surface of each platter. The heads are attached to a disk arm [11] . The surface of the platter is divided into tracks, which are subdivided into sectors. The set of tracks that are at one arm position called cylinder [2]. Once the head in a position, the read or write operation is then performed as the sector moves under the head [4].

In an attempt towards reducing the average seek time, our new algorithm is taking less number of head movement and less seek time compared to other disk scheduling algorithms.

1.1 *Disk Scheduling Criteria*

Generally a set of criteria is established against which various scheduling policies are evaluated.

- Seek Time: The time for the disk arm to move the heads to the cylinder containing the desired sector.
- Rotational latency: The additional time incurred for the disk to rotate the desired sector to the disk head.
- Disk bandwidth: Total number of bytes transferred divided by the total time between the first request for service and the completion of last transfer.
- Transfer time: The time required for the transfer. this depends on the rotational speed of the disk.

2. Disk Scheduling Algorithm

There are several disk scheduling algorithms such as FCFS, SSTF, SCAN, C-SCAN, LOOK and C-LOOK algorithm etc, which helps in scheduling the request.

- First Come First Served (FCFS): This algorithm simply service the request in FCFS basis, in which the earliest arriving request is serviced first if the load becomes heavy, FCFS can be long waiting times [3].
- Shortest Seek Time First (SSTF): This algorithm service the request with least seek time from the current head position before moving the head to service other request. This algorithm gives substantial improvement in performance compared to FCFS [11].
- SCAN/Elevator**:** In this algorithm, the disk arm starts at one end of the disk, servicing requests as it reaches each cylinder until it gets to the other end of the disk. Once it reaches the other end, reverse the direction of head movement and servicing continues.
- C-SCAN: This algorithm is the variant of SCAN algorithm only difference is that when the head reaches the other end, it immediately returns to the beginning of the disk without servicing any requests on a return trip.
- LOOK: This algorithm looks ahead to the end of the current sweep to determine the next request to service. if there are no more requests in the current direction, LOOK changes the preferred direction and begins the next sweep. This algorithm eliminated unnecessary seek operations exhibited by other

variations of the SCAN [3].

- C-LOOK: This algorithm is the variation of LOOK and uses the same technique as C-SCAN. When there is no more requests on a current inward sweep. The read / write head moves to the request closest to the outer cylinder and begins the next sweep. It results in high throughput compared to LOOK [3].

3. Related Work Done

In the recent years many researchers have came forward with their work and ideas of reducing the total number of head movement and minimizing seek time in the disk scheduling algorithm.

1. Sandipon saha, Md. Wasim Akhter, Mohd Abul Karhem, suggested a new real time disk scheduler that reduces the head movement [5].

2. Sourav Kumar Bhoi, Sanjaya Kumar panda, Imran Hossain Farak, proposed a ODSA algorithm which increases the efficiency of the disk performance by reducing seek time and transfer time [2].

3. Priya hooda, Supriya Raheja, proposed a FDS algorithm that improves rotational delay compared to other algorithms [6].

4. Kitae Hwang, Heoushik Shin, proposed two disk scheduling algorithm of SRLF and SATF for reduced rotational latency are more efficient than the existing algorithms [12].

5. Zoran Dimitrijevic, Raju Rangaswami, Edward Y Chang, proposed a preemption mechanisms show that semi-preemptible i/o improved the preemptibility of disk access with little loss in disk throughput [7].

6. Mathew Andrews, Michael A Bender, Lisa Zhang, proposed new disk scheduling algorithm using 3/2 approximation algorithm, NP hardness proof, optimal algorithm for linear reachability functions [8].

7. Hu Ming, proposed a method based on the idea of disk arm and rotational position and showed increase in the disk rotation leads to higher data transfer time [9].

8. David M Jacobson and John Wilkes, proposed a new goal of minimizing overall access time taking into account rotational position will improve disk bandwidth [1].

9. Margo Seltzer, Peter Chen, John Ousterhunt, proposed GSTF and WSTF scheduling mechanism to improve the performance of disk scheduling algorithm [10].

4. Proposed Algorithm

The main goal of our proposed algorithm (SMCC) is to minimize the number of head movement and seek time compared to other algorithms thereby improving the performance of our algorithm.

4.1 Sort Mid Current Comparison (SMCC) Disk Scheduling Algorithm

Assuming that the disk controller and disk drive are busy doing something. The request that can't be serviced by the hardware as soon as arrived is placed in the queue of pending requests. Assuming that the requests are in the random order. Now apply any sorting method to sort the requests in the ascending order and then obtain the midpoint request from the sorted queue. Once we know the midpoint request, then we will compare the current head pointer with the midpoint request. If the current head pointer is less than the midpoint request the we will

service the requests one by one from initial request until we reach the last request in the sorted list. Else we will service the requests one by one from the last request until we reach the initial request in the sorted list. In either case the scanning should be done from the current head pointer. Finally we will calculate the total number of head movement and average seek time.

The pseudocode of our proposed algorithm (SMCC Disk Scheduling Algorithm) is shown below.

1. Declaration and Initialization
   A [ ] // the list of pending requests which are waiting to be serviced. It is of type int
   N // Total number of pending waiting requests in A [ ]. Initially it is 0 and is of type int
   CHP // Current Head Position
   THM // Total number of Head Movement. Initially it is 0 and is of type int
   AST // Average Seek Time. It is of type float
   MPR // Mid Point Request obtained from the sorted array. It is of type int
   low // indicates the index of the starting element in A [ ]
   high // indicates the index of the last element in A [ ]
2. Use any sorting method to sort the pending requests in the unsorted array A [ ].
3. After sorting the array, obtain the Mid Point Request from the sorted array A [ ]
   MPR = (low + high) / 2 // this will be needed for our future comparisons
4. Read the CHP
   CHP = Initial disk head position
5. for i = 0 to N-1 do {
   if (CHP < MPR) {
   Scanning will start from A [i] up to the last element A [i-1] from the CHP
   // Calculate Total no. of Head Movement and Average Seek Time
   THM = CHP + | $A_{i-1} - A_i$ |
   AST = THM / N
   } // end if
   else if (CHP > MPR) {
   Scanning will start from last element A [i-1] to A [i] from the CHP
   // Calculate Total no. of Head Movement and Average Seek Time
   THM = CHP + | $A_{i-1} - A_I$ |
   AST = THM / N
   } // else if
   break;
   } // end for
6. Stop the algorithm

5. Results and Analysis

Before we implement the steps performed in the proposed method, we have taken three different cases to demonstrate the general disk scheduling algorithm without proposed method. In each case, we have taken different pending requests and we will calculate the total head movement and average seek time for each algorithm. In the next section, we will implement the proposed algorithm and we will compare the results obtained in these three cases with our new algorithm to show the improved performance.

Case 1: Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with requests for i/o to blocks on cylinder : 98, 183, 37, 122, 14, 124, 65, 67. Assume that disk head is currently at cylinder 53. figure 1 to figure 6 show the representation of FCFS, SSTF, SSTF, SCAN, C-SCAN, LOOK and C-LOOK disk

scheduling algorithm respectively.

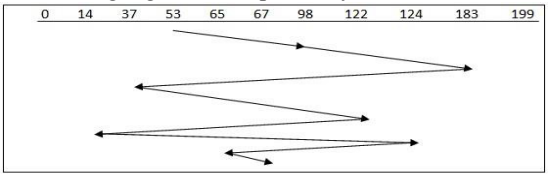

Fig. 1 : Representation of FCFS for CASE 1

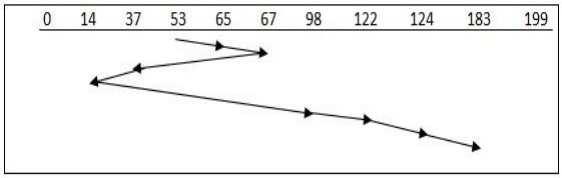

Fig. 2 : Representation of SSTF for CASE 1

Total head movement= (98-53)+(183-98)+(183-37)
+(122-37)+(122-14)+(124-14)+(124-65)+(67-65)
Total head movement = 640 cylinders
Average Seek Time = 640/8 = 80

Total head movement = (65-53)+(67-65)+(67-37)+
(37-14)+(98-14)+(122-98)+(124-122)+(183-124)
Total head movement = 236 cylinders
Average Seek Time = 236/8 = 29.5

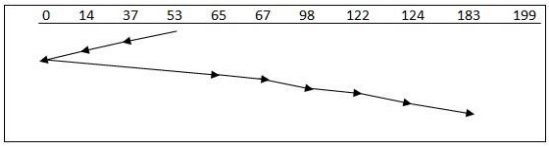

Fig. 3: Representation of SCAN for CASE 1

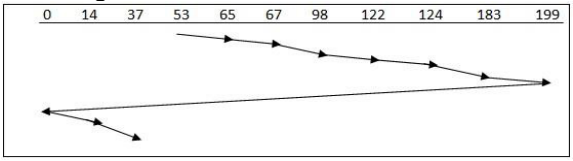

Fig. 4: Representation of C-SCAN for CASE 1

Total head movement = (53-37)+(37-14)+(14-0)+
(65-0)+(67-65)+(98-67)+(122-98)+(124-122)+(183
-124) = 236 Cylinders
Average Seek Time = 236/8 = 29.5

Total head movement = (65-53)+(67-65)+(98-67)+
(122-98)+(124-122)+(183-124)+(199-183)+
(199-0)+(14-0)+(37-14) = 382 Cylinders
Average Seek Time = 382/8 = 47.75

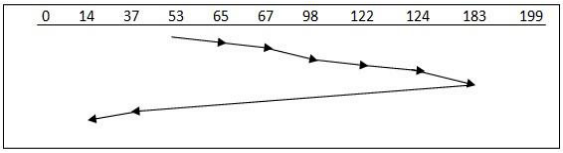

Fig. 5: Representation of LOOK for CASE 1

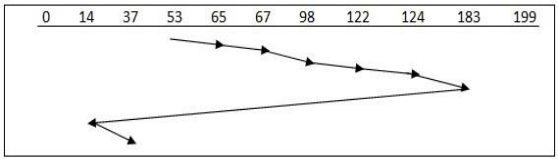

Fig. 6: Representation of C-LOOK for CASE 1

Total head movement = (65-53)+(67-65)+(98-67)
+(122-98)+(124-122)+(183-124)+(183-37)+(37-14)
Total head movement = 299 Cylinders
Average Seek Time = 299/8 = 37.37

Total head movement = (65-53)+(67-65)+(98-67)
+(122-98)+(124-122)+(183-124)+(183-14)+(37-
14) = 322 Cylinders
Average Seek Time = 322/8 = 40.25

Case 2 : Suppose a disk drive has 100 cylinders, numbered 0 to 99. Consider a disk queue with requests for i/o to blocks on cylinder : 33, 72, 47, 8, 99, 74, 52, 75. Assume that disk head is currently at cylinder 63. figure 7 to figure 12 show the representation of FCFS, SSTF, SSTF, SCAN, C-SCAN, LOOK and C-LOOK disk scheduling algorithm respectively.

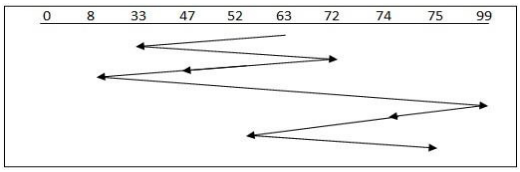

Fig. 7 : Representation of FCFS for CASE 2

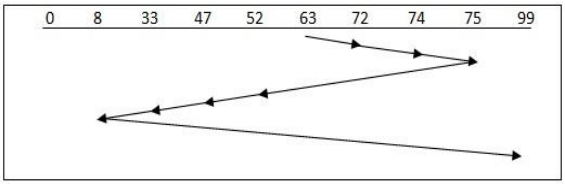

Fig. 8 : Representation of SSTF for CASE 2

Total head movement = (63-33)+(72-33)+(72-47)

Total head movement = (72-63)+(74-72)+(75-74)+

+(47-8)+(99-8)+(99-74)+(74-52)+(75-52)
Total head movement = 294 Cylinders
Average Seek Time = 294/8 = 36.75

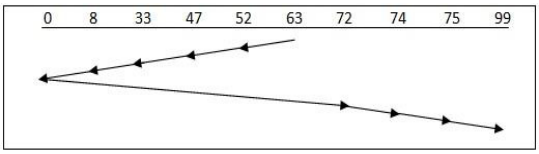

Fig. 9 : Representation of SCAN for CASE 2

(75-52)+(52-47)+(47-33)+(33-8)+(99-8)
Total head movement = 170 Cylinders
Average Seek Time = 170/8 = 21.25

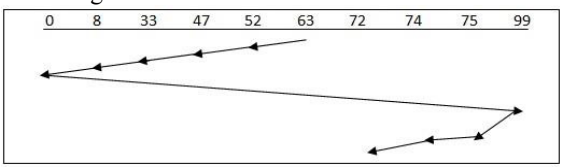

Fig. 10 : Representation of C-SCAN for CASE 2

Total head movement = (63-52)+(52-47)+(47-33)+
(33-8)+(8-0)+(72-0)+(74-72)+(75-74)+(99-75)
Total head movement = 162 Cylinders
Average Seek Time = 162/8 = 20.25

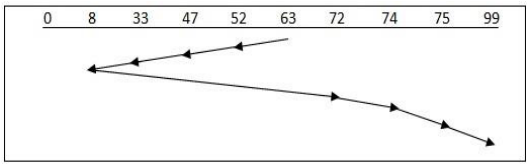

Fig. 11 : Representation of LOOK for CASE 2

Total head movement = (63-52)+(52-47)+(47-33)+
(33-8)+(8-0)+(99-0)+(99-75)+(75-74)+(74-72)
Total head movement = 189 Cylinders
Average Seek Time = 189/8 = 23.62

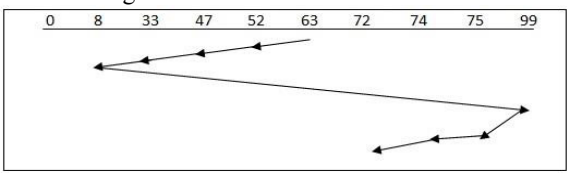

Fig. 12 : Representation of C-LOOK for CASE 2

Total head movement = (63-52)+(52-47)+(47-33)+
(33-8)+(72-8)+(74-72)+(75-74)+(99-75)
Total head movement = 146 Cylinders
Average Seek Time = 18.25

Total head movement = (63-52)+(52-47)+(47-33)+
(33-8)+(99-8)+(99-75)+(75-74)+(74-72)
Total head movement = 173 Cylinders
Average Seek Time = 173/8 = 21.62

Case 3 : Suppose a disk drive has 5000 cylinders, numbered 0 to 4999. Consider a disk queue with requests for i/o to blocks on cylinder : 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130 . Assume that disk head is currently at cylinder 143. And the previous request was at cylinder 125. Figure 13 to Figure 18 show the representation of FCFS, SSTF, SSTF, SCAN, C-SCAN, LOOK and C-LOOK disk scheduling algorithm respectively.

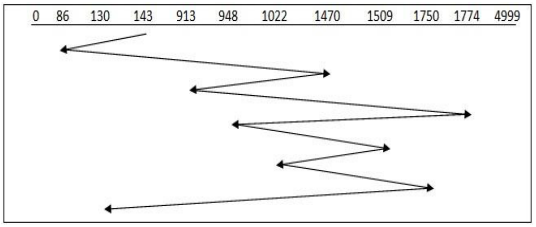

Fig. 13 : Representation of FCFS for CASE 3

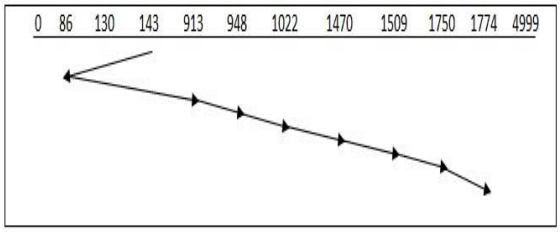

Fig. 14 : Representation of SSTF for CASE 3

Total head movement = (143-86)+(1470-86)+(1470
-913)+(1774-913)+(1774-948)+(1509-948)+(1509
-1022)+(1750-1022)+(1750-130)
Total head movement = 7081 Cylinders
Average Seek Time = 7081/9 = 786.77

Total head movement = (143-130)+(130-86)+(913-86)
+(948-913)+(1022-948)+(1470-1022)+(1509-1470)+
(1750-1509)+(1774-1750)
Total head movement = 1745 Cylinders
Average Seek Time = 1745/9 = 193.88

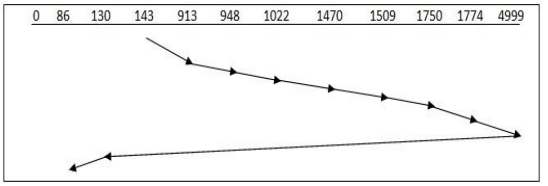

Fig. 15 : Representation of SCAN for CASE 3

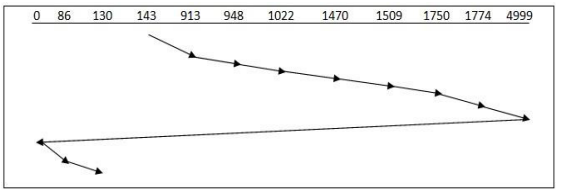

Fig. 16 : Representation of C-SCAN for CASE 3

Total head movement = (913-143)+(948-913)+(1022 -948)+(1470-1022)+(1509-1470)+(1750-1509)+ (1774-1750)+(4999-1774)+(4999-130)+(130-86)
Total head movement = 9769 Cylinders
Average Seek Time = 9769/9 = 1085.44

Total head movement = (913-143)+(948-913)+(1022- 948)+(1470-1022)+(1509-1470)+(1750-1509)+(1774 -1750)+(4999-1774)+(4999-0)+(86-0)+(130-86)
Total head movement = 9985 Cylinders
Average Seek Time = 9985/9 = 1109.44

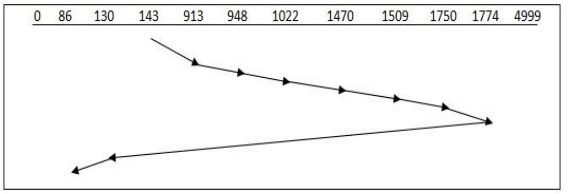

Fig. 17 : Representation of LOOK for CASE 3

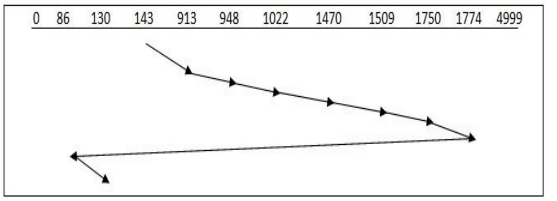

Fig. 18 : Representation of C-LOOK for CASE 3

Total head movement = (913-143)+(948-913)+(1022 -948)+(1470-1022)+(1509-1470)+(1750-1509)+ (1774-1750)+(1774-130)+(130-86)
Total head movement = 3319 Cylinders
Average Seek Time = 3319/9 = 368.77

Total head movement = (913-143)+(948-913)+ (1022- 948)+(1470-1022)+(1509-1470)+(1750-1509) (1774-1750)+(1774-86)+(130-86)
Total head movement = 3363 Cylinders
Average Seek Time = 3363/9 = 373.66

5.1 Proposed Method

In this section we will apply the proposed algorithm for the three different requests obtained from the above cases to show how the requests can be accessed faster.

Case 1 : Suppose a disk drive has 200 cylinders, numbered 0 to 199. Consider a disk queue with requests for i/o to blocks on cylinder : 98, 183, 37, 122, 14, 124, 65, 67. Assume that disk head is currently at cylinder 53. Figure 19 show the representation of Sort Mid Current Comparison (SMCC) disk scheduling algorithm. Table 1 show the comparison of all algorithms with our new algorithm.

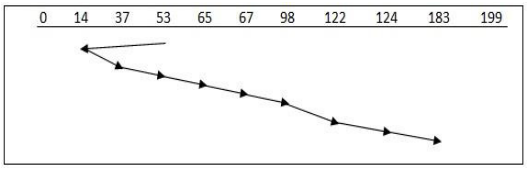

Fig. 19 : Representation of SMCC DSK for CASE 1

Total head movement = (53-14)+(37-14)+(65-37)+ (67-65)+(98-67)+(122-98)+(124-122)+(183-124)
Total head movement = 199 Cylinders
Average Seek Time (AST) = 199 / 8 = 24.87

Table 1. Comparisons of all the algorithms with SMCC for CASE 1

Algorithms	**Average Seek Time (AST)**	**Total Head Movement (THM)**
FCFS	80	640
SSTF	29.5	236

SCAN	29.5	236
C-SCAN	47.75	382
LOOK	37.37	299
C-LOOK	40.25	322
SMCC	24.87	199

Case 2 : Suppose a disk drive has 100 cylinders, numbered 0 to 99. Consider a disk queue with requests for i/o to blocks on cylinder : 33, 72, 47, 8, 99, 74, 52, 75. Assume that disk head is currently at cylinder 63. Figure 20 show the representation of Sort Mid Current Comparison (SMCC) disk scheduling algorithm. Table 2 show the comparison of all algorithms with our new algorithm.

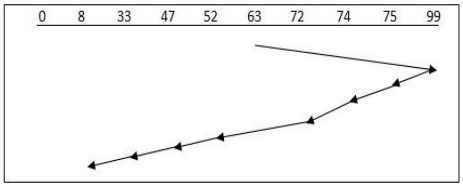

Fig. 20 : Representation of SMCC DSK for CASE 2

Total head movement = (99-63)+(99-75)+(75-74)+ (74-72) +(72-52)+(52-47)+(47-33)+(32-8)
Total head movement = 126 Cylinders
Average Seek Time = 126 / 8 = 15.75

Table 2. Comparisons of all the algorithms with SMCC for CASE 2

Algorithms	**Average Seek Time (AST)**	**Total Head Movement (THM)**
FCFS	36.75	294
SSTF	21.25	170
SCAN	20.25	162
C-SCAN	23.62	189
LOOK	18.25	146
C-LOOK	21.62	173
SMCC	15.75	126

Case 3 : Suppose a disk drive has 5000 cylinders, numbered 0 to 4999. Consider a disk queue with requests for i/o to blocks on cylinder : 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130 . Assume that disk head is currently at cylinder 143. Figure 21 show the representation of SMCC disk scheduling algorithm. Table 3 show the comparison of all algorithms with our new algorithm.

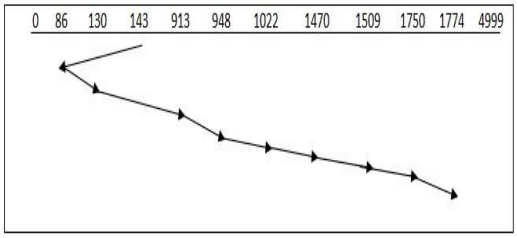

Fig. 21 : Representation of SMCC DSK for CASE 3

Total head movement = (143-86)+(130-86)+(913-130)+(948-913) +(1022-948)+(1470-1022)+(1509-1470) +(1750-1509)+(1774-1750)
Total head movement = 1745 Cylinders
Average Seek Time = 1745 / 9 = 193.88

Table 3. Comparisons of all the algorithms with SMCC for CASE 3

Algorithms	Average Seek Time (AST)	Total Head Movement (THM)
FCFS	786.77	7081
SSTF	193.88	1745
SCAN	1085.44	9769
C-SCAN	1109.44	9985
LOOK	368.77	3319
C-LOOK	373.66	3363
SMCC	193.88	1745

5.3 Comparisons

Figure 22 shows the comparisons of Total no. of head movement of and Averege seek time of all the algorithms with our new algorithm called SMCC disk scheduling algorithm for CASE 1. Thus from this figure we showed that our new algorithm will results in reduced seek time and head movement compared to other disk scheduling algorithms.

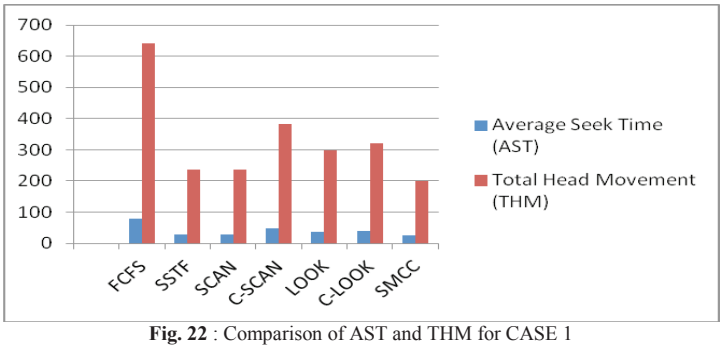

Fig. 22 : Comparison of AST and THM for CASE 1

Figure 23 shows the comparisons of Total no. of head movement of and Average seek time of all the algorithms with our new algorithm called SMC disk scheduling algorithm for CASE 2. Thus from this figure we showed that our new algorithm will results in reduced seek time and head movement compared to other disk scheduling algorithms.

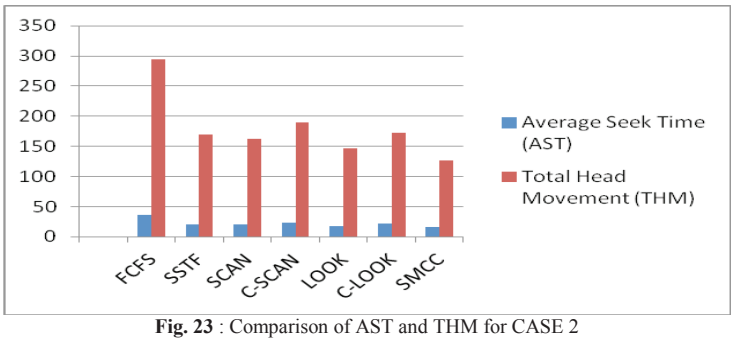

Fig. 23 : Comparison of AST and THM for CASE 2

Figure 24 shows the comparisons of Total no. of head movement of and Average seek time of all the algorithms with our new algorithm called SMC disk scheduling algorithm for CASE 3. Thus from this figure we showed that our new algorithm's THM and AST is equal to SSTF disk scheduling algorithm.

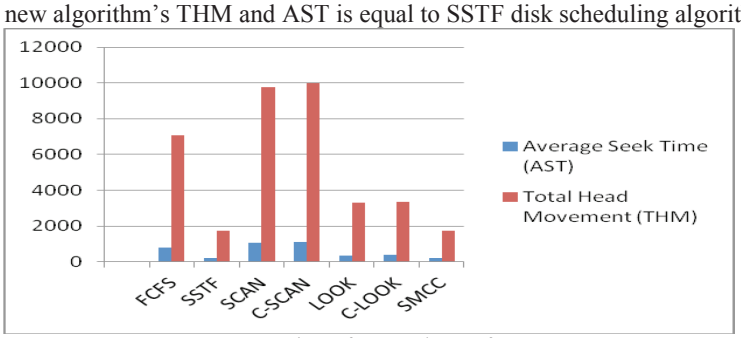

Fig. 24 : Comparison of AST and THM for CASE 3

6. Conclusion

The performance of disk scheduling algorithm depends heavily on the total number of head movement, seek time and rotational latency. With the classical approach of disk scheduling algorithm, few algorithms like SSTF and LOOK will be the most efficient algorithm compared to FCFS, SCAN, C-SCAN and C-LOOK disk scheduling algorithm with respect to these parameters.

Based on our experiment, we have proposed a new algorithm called Sort Mid Current Comparison (SMCC) disk scheduling algorithm. Compared to the classical approach of disk scheduling algorithm, our results and calculations show that our proposed algorithm reduces the number of head movement and seek time thus improving the performance of disk bandwidth for disk drives. For few requests, our algorithm is equal to SSTF / LOOK disk scheduling algorithm.

References

[1] D. M. Jacobson and J. Wilkes, "Disk Scheduling Algorithms Based on Rotational Position," *Technical Report*, 1991.
[2] Sourav Kumar Bhoi, Sanjaya Kumar Panda, Imran Hossain Faruk, "Design and Performance Evaluation of an Optimized Disk Scheduling Algorithm (ODSA)", International Journal of Computer Applications (0975 – 8887) Volume 40– No.11, February 2012.
[3] Dietel, Dietel and Choffnes, Operating Systems, 3rd edition, Pearson education, 2009
[4] William Stallings, "Operating Systems: Internal and Design Principles", seventh edition, prentice hall, 2012.
[5] S. Saha, N. Akhter and M. A. Kashem, "A New Heuristic Disk Scheduling Algorithm," *International Journal of Scientific & Technology and Research*, Vol. 2, 2013, pp. 49-53.
[6] Priya Hooda, Supriya Raheja, "A New Approach to Disk Scheduling Using Fuzzy Logic", *Journal of Computer and Communications*, 2014, 2, 1-5.
[7] Z. Dimitrijevic, R. Rangaswami and E. Y. Chang, "Support for Preemptive Disk Scheduling", IEEE Transactions on computers, Vol. 54, No. 10, Oct 2005.
[8] M. Andrews, M. A. Bender and L. Zhang, "New Algorithms for Disk Scheduling".
[9] Hu. Ming, "Improved Disk Scheduling Algorithms Based on Rotational Position," *Journal of Shanghai University*, Vol. 9, No. 5, 2005, pp. 411-414.
[10] M. Seltzer, P. Chen and J. Ousterhout, "Disk Scheduling Revisited," *Proceedings of the* 1990 *Winter Usenix*, Washington DC, 1990.
[11] Silberchatz, Galvin and Gagne, operating systems concepts, 8th edition, John Wiley and Sons, 2012
[12] Kitae Hwang and Heonshik Shin, "New Disk Scheduling Algorithms for Reduced Rotational Latency"